Network Working Group   M. Murata
Request for Comments: 3023   IBM Tokyo Research Laboratory
Obsoletes: 2376   S. St.Laurent
Updates: 2048   simonstl.com
Category: Standards Track   D. Kohn
                                                     Skymoon Ventures
                                                        January 2001

                           XML Media Types

Status of this Memo

Copyright Notice

Abstract

   This document standardizes five new media types -- text/xml,
   application/xml, text/xml-external-parsed-entity, application/xml-
   external-parsed-entity, and application/xml-dtd -- for use in
   exchanging network entities that are related to the Extensible Markup
   Language (XML). This document also standardizes a convention (using
   the suffix '+xml') for naming media types outside of these five types
   when those media types represent XML MIME (Multipurpose Internet Mail
   Extensions) entities. XML MIME entities are currently exchanged via
   the HyperText Transfer Protocol on the World Wide Web, are an
   integral part of the WebDAV protocol for remote web authoring, and
   are expected to have utility in many domains.

   Major differences from RFC 2376 are (1) the addition of text/xml-
   external-parsed-entity, application/xml-external-parsed-entity, and
   application/xml-dtd, (2) the '+xml' suffix convention (which also
   updates the RFC 2048 registration process), and (3) the discussion of
   "utf-16le" and "utf-16be".

Table of Contents

1. Introduction

   The World Wide Web Consortium has issued Extensible Markup Language
   (XML) 1.0 (Second Edition)[XML]. To enable the exchange of XML
   network entities, this document standardizes five new media types --
   text/xml, application/xml, text/xml-external-parsed-entity,
   application/xml-external-parsed-entity, and application/xml-dtd -- as
   well as a naming convention for identifying XML-based MIME media
   types.

   XML entities are currently exchanged on the World Wide Web, and XML
   is also used for property values and parameter marshalling by the
   WebDAV[RFC2518] protocol for remote web authoring. Thus, there is a
   need for a media type to properly label the exchange of XML network
   entities.

   Although XML is a subset of the Standard Generalized Markup Language
   (SGML) ISO 8879[SGML], which has been assigned the media types
   text/sgml and application/sgml, there are several reasons why use of
   text/sgml or application/sgml to label XML is inappropriate. First,
   there exist many applications that can process XML, but that cannot
   process SGML, due to SGML's larger feature set. Second, SGML
   applications cannot always process XML entities, because XML uses
   features of recent technical corrigenda to SGML. Third, the
   definition of text/sgml and application/sgml in [RFC1874] includes
   parameters for SGML bit combination transformation format (SGML-
   bctf), and SGML boot attribute (SGML-boot). Since XML does not use
   these parameters, it would be ambiguous if such parameters were given
   for an XML MIME entity. For these reasons, the best approach for
   labeling XML network entities is to provide new media types for XML.

   Since XML is an integral part of the WebDAV Distributed Authoring
   Protocol, and since World Wide Web Consortium Recommendations have
   conventionally been assigned IETF tree media types, and since similar
   media types (HTML, SGML) have been assigned IETF tree media types,
   the XML media types also belong in the IETF media types tree.

   Similarly, XML will be used as a foundation for other media types,
   including types in every branch of the IETF media types tree. To
   facilitate the processing of such types, media types based on XML,
   but that are not identified using text/xml or application/xml, SHOULD
   be named using a suffix of '+xml' as described in Section 7. This
   will allow XML-based tools -- browsers, editors, search engines, and
   other processors -- to work with all XML-based media types.

2. Notational Conventions

   The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
   "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
   document are to be interpreted as described in [RFC2119].

   As defined in [RFC2781], the three charsets "utf-16", "utf-16le", and
   "utf-16be" are used to label UTF-16 text. In this document, "the
   UTF-16 family" refers to those three charsets. By contrast, the
   phrases "utf-16" or UTF-16 in this document refer specifically to the
   single charset "utf-16".

   As sometimes happens between two communities, both MIME and XML have
   defined the term entity, with different meanings. Section 2.4 of
   [RFC2045] says:

      "The term 'entity' refers specifically to the MIME-defined header
      fields and contents of either a message or one of the parts in the
      body of a multipart entity."

Section 4 of [XML] says:

> "An XML document may consist of one or many storage units" called
> entities that "have content" and are normally "identified by
> name".

In this document, "XML MIME entity" is defined as the latter (an XML
entity) encapsulated in the former (a MIME entity).

3. XML Media Types

This document standardizes five media types related to XML MIME
entities: text/xml, application/xml, text/xml-external-parsed-entity,
application/xml-external-parsed-entity, and application/xml-dtd.
Registration information for these media types is described in the
sections below.

Within the XML specification, XML MIME entities can be classified
into four types. In the XML terminology, they are called "document
entities", "external DTD subsets", "external parsed entities", and
"external parameter entities". The media types text/xml and
application/xml MAY be used for "document entities", while text/xml-
external-parsed-entity or application/xml-external-parsed-entity
SHOULD be used for "external parsed entities". The media type
application/xml-dtd SHOULD be used for "external DTD subsets" or
"external parameter entities". application/xml and text/xml MUST NOT
be used for "external parameter entities" or "external DTD subsets",
and MUST NOT be used for "external parsed entities" unless they are
also well-formed "document entities" and are referenced as such.
Note that [RFC2376] (which this document obsoletes) allowed such
usage, although in practice it is likely to have been rare.

Neither external DTD subsets nor external parameter entities parse as
XML documents, and while some XML document entities may be used as
external parsed entities and vice versa, there are many cases where
the two are not interchangeable. XML also has unparsed entities,
internal parsed entities, and internal parameter entities, but they
are not XML MIME entities.

If an XML document -- that is, the unprocessed, source XML document
-- is readable by casual users, text/xml is preferable to
application/xml. MIME user agents (and web user agents) that do not
have explicit support for text/xml will treat it as text/plain, for
example, by displaying the XML MIME entity as plain text.
Application/xml is preferable when the XML MIME entity is unreadable
by casual users. Similarly, text/xml-external-parsed-entity is

preferable when an external parsed entity is readable by casual
users, but application/xml-external-parsed-entity is preferable when
a plain text display is inappropriate.

   NOTE: Users are in general not used to text containing tags such
   as <price>, and often find such tags quite disorienting or
   annoying. If one is not sure, the conservative principle would
   suggest using application/* instead of text/* so as not to put
   information in front of users that they will quite likely not
   understand.

The top-level media type "text" has some restrictions on MIME
entities and they are described in [RFC2045] and [RFC2046]. In
particular, the UTF-16 family, UCS-4, and UTF-32 are not allowed
(except over HTTP[RFC2616], which uses a MIME-like mechanism). Thus,
if an XML document or external parsed entity is encoded in such
character encoding schemes, it cannot be labeled as text/xml or
text/xml-external-parsed-entity (except for HTTP).

Text/xml and application/xml behave differently when the charset
parameter is not explicitly specified. If the default charset (i.e.,
US-ASCII) for text/xml is inconvenient for some reason (e.g., bad web
servers), application/xml provides an alternative (see "Optional
parameters" of application/xml registration in Section 3.2). The
same rules apply to the distinction between text/xml-external-
parsed-entity and application/xml-external-parsed-entity.

XML provides a general framework for defining sequences of structured
data. In some cases, it may be desirable to define new media types
that use XML but define a specific application of XML, perhaps due to
domain-specific security considerations or runtime information.
Furthermore, such media types may allow UTF-8 or UTF-16 only and
prohibit other charsets. This document does not prohibit such media
types and in fact expects them to proliferate. However, developers
of such media types are STRONGLY RECOMMENDED to use this document as
a basis for their registration. In particular, the charset parameter
SHOULD be used in the same manner, as described in Section 7.1, in
order to enhance interoperability.

An XML document labeled as text/xml or application/xml might contain
namespace declarations, stylesheet-linking processing instructions
(PIs), schema information, or other declarations that might be used
to suggest how the document is to be processed. For example, a
document might have the XHTML namespace and a reference to a CSS
stylesheet. Such a document might be handled by applications that
would use this information to dispatch the document for appropriate
processing.

3.1 Text/xml Registration

   MIME media type name: text

   MIME subtype name: xml

   Mandatory parameters: none

   Optional parameters: charset

      Although listed as an optional parameter, the use of the charset
      parameter is STRONGLY RECOMMENDED, since this information can be
      used by XML processors to determine authoritatively the character
      encoding of the XML MIME entity. The charset parameter can also
      be used to provide protocol-specific operations, such as charset-
      based content negotiation in HTTP. "utf-8" [RFC2279] is the
      recommended value, representing the UTF-8 charset. UTF-8 is
      supported by all conforming processors of [XML].

      If the XML MIME entity is transmitted via HTTP, which uses a
      MIME-like mechanism that is exempt from the restrictions on the
      text top-level type (see section 19.4.1 of [RFC2616]), "utf-16"
      [RFC2781]) is also recommended. UTF-16 is supported by all
      conforming processors of [XML]. Since the handling of CR, LF and
      NUL for text types in most MIME applications would cause undesired
      transformations of individual octets in UTF-16 multi-octet
      characters, gateways from HTTP to these MIME applications MUST
      transform the XML MIME entity from text/xml; charset="utf-16" to
      application/xml; charset="utf-16".

      Conformant with [RFC2046], if a text/xml entity is received with
      the charset parameter omitted, MIME processors and XML processors
      MUST use the default charset value of "us-ascii"[ASCII]. In cases
      where the XML MIME entity is transmitted via HTTP, the default
      charset value is still "us-ascii". (Note: There is an
      inconsistency between this specification and HTTP/1.1, which uses
      ISO-8859-1[ISO8859] as the default for a historical reason. Since
      XML is a new format, a new default should be chosen for better
      I18N. US-ASCII was chosen, since it is the intersection of UTF-8
      and ISO-8859-1 and since it is already used by MIME.)

      There are several reasons that the charset parameter is
      authoritative. First, some MIME processing engines do transcoding
      of MIME bodies of the top-level media type "text" without
      reference to any of the internal content. Thus, it is possible
      that some agent might change text/xml; charset="iso-2022-jp" to
      text/xml; charset="utf-8" without modifying the encoding
      declaration of an XML document. Second, text/xml must be

compatible with text/plain, since MIME agents that do not
understand text/xml will fallback to handling it as text/plain.
If the charset parameter for text/xml were not authoritative, such
fallback would cause data corruption. Third, recent web servers
have been improved so that users can specify the charset
parameter. Fourth, [RFC2130] specifies that the recommended
specification scheme is the "charset" parameter.

Since the charset parameter is authoritative, the charset is not
always declared within an XML encoding declaration. Thus, special
care is needed when the recipient strips the MIME header and
provides persistent storage of the received XML MIME entity (e.g.,
in a file system). Unless the charset is UTF-8 or UTF-16, the
recipient SHOULD also persistently store information about the
charset, perhaps by embedding a correct XML encoding declaration
within the XML MIME entity.

Encoding considerations: This media type MAY be encoded as
appropriate for the charset and the capabilities of the underlying
MIME transport. For 7-bit transports, data in UTF-8 MUST be
encoded in quoted-printable or base64. For 8-bit clean transport
(e.g., 8BITMIME[RFC1652] ESMTP or NNTP[RFC0977]), UTF-8 does not
need to be encoded. Over HTTP[RFC2616], no content-transfer-
encoding is necessary and UTF-16 may also be used.

Security considerations: See Section 10.

Interoperability considerations: XML has proven to be interoperable
across WebDAV clients and servers, and for import and export from
multiple XML authoring tools. For maximum interoperability,
validating processors are recommended. Although non-validating
processors may be more efficient, they are not required to handle
all features of XML. For further information, see sub-section 2.9
"Standalone Document Declaration" and section 5 "Conformance" of
[XML].

Published specification: Extensible Markup Language (XML) 1.0 (Second
Edition)[XML].

Applications which use this media type: XML is device-, platform-,
and vendor-neutral and is supported by a wide range of Web user
agents, WebDAV[RFC2518] clients and servers, as well as XML
authoring tools.

Additional information:

Magic number(s): None.

Although no byte sequences can be counted on to always be
present, XML MIME entities in ASCII-compatible charsets
(including UTF-8) often begin with hexadecimal 3C 3F 78 6D 6C
("<?xml"), and those in UTF-16 often begin with hexadecimal FE
FF 00 3C 00 3F 00 78 00 6D 00 6C or FF FE 3C 00 3F 00 78 00 6D
00 6C 00 (the Byte Order Mark (BOM) followed by "<?xml"). For
more information, see Appendix F of [XML].

File extension(s): .xml

Macintosh File Type Code(s): "TEXT"

Person and email address for further information:

MURATA Makoto (FAMILY Given) <mmurata@trl.ibm.co.jp>

Simon St.Laurent <simonstl@simonstl.com>

Daniel Kohn <dan@dankohn.com>

Intended usage: COMMON

Author/Change controller: The XML specification is a work product of
the World Wide Web Consortium's XML Working Group, and was edited
by:

Tim Bray <tbray@textuality.com>

Jean Paoli <jeanpa@microsoft.com>

C. M. Sperberg-McQueen <cmsmcq@uic.edu>

Eve Maler <eve.maler@east.sun.com>

The W3C, and the W3C XML Core Working Group, have change control
over the XML specification.

3.2 Application/xml Registration

MIME media type name: application

MIME subtype name: xml

Mandatory parameters: none

   Optional parameters: charset

      Although listed as an optional parameter, the use of the charset
      parameter is STRONGLY RECOMMENDED, since this information can be
      used by XML processors to determine authoritatively the charset of
      the XML MIME entity. The charset parameter can also be used to
      provide protocol-specific operations, such as charset-based
      content negotiation in HTTP.

      "utf-8" [RFC2279] and "utf-16" [RFC2781] are the recommended
      values, representing the UTF-8 and UTF-16 charsets, respectively.
      These charsets are preferred since they are supported by all
      conforming processors of [XML].

      If an application/xml entity is received where the charset
      parameter is omitted, no information is being provided about the
      charset by the MIME Content-Type header. Conforming XML
      processors MUST follow the requirements in section 4.3.3 of [XML]
      that directly address this contingency. However, MIME processors
      that are not XML processors SHOULD NOT assume a default charset if
      the charset parameter is omitted from an application/xml entity.

      There are several reasons that the charset parameter is
      authoritative. First, recent web servers have been improved so
      that users can specify the charset parameter. Second, [RFC2130]
      specifies that the recommended specification scheme is the
      "charset" parameter.

      On the other hand, it has been argued that the charset parameter
      should be omitted and the mechanism described in Appendix F of
      [XML] (which is non-normative) should be solely relied on. This
      approach would allow users to avoid configuration of the charset
      parameter; an XML document stored in a file is likely to contain a
      correct encoding declaration or BOM (if necessary), since the
      operating system does not typically provide charset information
      for files. If users would like to rely on the encoding
      declaration or BOM and to hide charset information from protocols,
      they may determine not to use the parameter.

      Since the charset parameter is authoritative, the charset is not
      always declared within an XML encoding declaration. Thus, special
      care is needed when the recipient strips the MIME header and
      provides persistent storage of the received XML MIME entity (e.g.,
      in a file system). Unless the charset is UTF-8 or UTF-16, the
      recipient SHOULD also persistently store information about the
      charset, perhaps by embedding a correct XML encoding declaration
      within the XML MIME entity.

Encoding considerations: This media type MAY be encoded as
   appropriate for the charset and the capabilities of the underlying
   MIME transport. For 7-bit transports, data in either UTF-8 or
   UTF-16 MUST be encoded in quoted-printable or base64. For 8-bit
   clean transport (e.g., 8BITMIME[RFC1652] ESMTP or NNTP[RFC0977]),
   UTF-8 is not encoded, but the UTF-16 family MUST be encoded in
   base64. For binary clean transports (e.g., HTTP[RFC2616]), no
   content-transfer-encoding is necessary.

Security considerations: See Section 10.

Interoperability considerations: Same as Section 3.1.

Published specification: Same as Section 3.1.

Applications which use this media type: Same as Section 3.1.

Additional information: Same as Section 3.1.

Person and email address for further information: Same as Section
   3.1.

Intended usage: COMMON

Author/Change controller: Same as Section 3.1.

3.3 Text/xml-external-parsed-entity Registration

MIME media type name: text

MIME subtype name: xml-external-parsed-entity

Mandatory parameters: none

Optional parameters: charset

   The charset parameter of text/xml-external-parsed-entity is
   handled the same as that of text/xml as described in Section 3.1.

Encoding considerations: Same as Section 3.1.

Security considerations: See Section 10.

Interoperability considerations: XML external parsed entities are as
   interoperable as XML documents, though they have a less tightly
   constrained structure and therefore need to be referenced by XML
   documents for proper handling by XML processors. Similarly, XML
   documents cannot be reliably used as external parsed entities

because external parsed entities are prohibited from having
standalone document declarations or DTDs. Identifying XML
external parsed entities with their own content type should
enhance interoperability of both XML documents and XML external
parsed entities.

Published specification: Same as Section 3.1.

Applications which use this media type: Same as Section 3.1.

Additional information:

Magic number(s): Same as Section 3.1.

File extension(s): .xml or .ent

Macintosh File Type Code(s): "TEXT"

Person and email address for further information: Same as Section
3.1.

Intended usage: COMMON

Author/Change controller: Same as Section 3.1.

3.4 Application/xml-external-parsed-entity Registration

MIME media type name: application

MIME subtype name: xml-external-parsed-entity

Mandatory parameters: none

Optional parameters: charset

The charset parameter of application/xml-external-parsed-entity is
handled the same as that of application/xml as described in
Section 3.2.

Encoding considerations: Same as Section 3.2.

Security considerations: See Section 10.

Interoperability considerations: Same as those for text/xml-
external-parsed-entity as described in Section 3.3.

Published specification: Same as text/xml as described in Section
3.1.

Applications which use this media type: Same as Section 3.1.

Additional information:

   Magic number(s): Same as Section 3.1.

   File extension(s): .xml or .ent

   Macintosh File Type Code(s): "TEXT"

Person and email address for further information: Same as Section
   3.1.

Intended usage: COMMON

Author/Change controller: Same as Section 3.1.

3.5 Application/xml-dtd Registration

MIME media type name: application

MIME subtype name: xml-dtd

Mandatory parameters: none

Optional parameters: charset

   The charset parameter of application/xml-dtd is handled the same
   as that of application/xml as described in Section 3.2.

Encoding considerations: Same as Section 3.2.

Security considerations: See Section 10.

Interoperability considerations: XML DTDs have proven to be
   interoperable by DTD authoring tools and XML browsers, among
   others.

Published specification: Same as text/xml as described in Section
   3.1.

Applications which use this media type: DTD authoring tools handle
   external DTD subsets as well as external parameter entities. XML
   browsers may also access external DTD subsets and external
   parameter entities.

Additional information:

    Magic number(s): Same as Section 3.1.

    File extension(s): .dtd or .mod

    Macintosh File Type Code(s): "TEXT"

Person and email address for further information: Same as Section
    3.1.

Intended usage: COMMON

Author/Change controller: Same as Section 3.1.

3.6 Summary

The following list applies to text/xml, text/xml-external-parsed-
entity, and XML-based media types under the top-level type "text"
that define the charset parameter according to this specification:

o Charset parameter is strongly recommended.

o If the charset parameter is not specified, the default is "us-
   ascii". The default of "iso-8859-1" in HTTP is explicitly
   overridden.

o No error handling provisions.

o An encoding declaration, if present, is irrelevant, but when
   saving a received resource as a file, the correct encoding
   declaration SHOULD be inserted.

The next list applies to application/xml, application/xml-external-
parsed-entity, application/xml-dtd, and XML-based media types under
top-level types other than "text" that define the charset parameter
according to this specification:

o Charset parameter is strongly recommended, and if present, it
   takes precedence.

o If the charset parameter is omitted, conforming XML processors
   MUST follow the requirements in section 4.3.3 of [XML].

4. The Byte Order Mark (BOM) and Conversions to/from the UTF-16 Charset

   Section 4.3.3 of [XML] specifies that XML MIME entities in the
   charset "utf-16" MUST begin with a byte order mark (BOM), which is a
   hexadecimal octet sequence 0xFE 0xFF (or 0xFF 0xFE, depending on
   endian). The XML Recommendation further states that the BOM is an
   encoding signature, and is not part of either the markup or the
   character data of the XML document.

   Due to the presence of the BOM, applications that convert XML from
   "utf-16" to a non-Unicode encoding MUST strip the BOM before
   conversion. Similarly, when converting from another encoding into
   "utf-16", the BOM MUST be added after conversion is complete.

   In addition to the charset "utf-16", [RFC2781] introduces "utf-16le"
   (little endian) and "utf-16be" (big endian) as well. The BOM is
   prohibited for these charsets. When an XML MIME entity is encoded in
   "utf-16le" or "utf-16be", it MUST NOT begin with the BOM but SHOULD
   contain an encoding declaration. Conversion from "utf-16" to "utf-
   16be" or "utf-16le" and conversion in the other direction MUST strip
   or add the BOM, respectively.

5. Fragment Identifiers

   Section 4.1 of [RFC2396] notes that the semantics of a fragment
   identifier (the part of a URI after a "#") is a property of the data
   resulting from a retrieval action, and that the format and
   interpretation of fragment identifiers is dependent on the media type
   of the retrieval result.

   As of today, no established specifications define identifiers for XML
   media types. However, a working draft published by W3C, namely "XML
   Pointer Language (XPointer)", attempts to define fragment identifiers
   for text/xml and application/xml. The current specification for
   XPointer is available at http://www.w3.org/TR/xptr.

6. The Base URI

   Section 5.1 of [RFC2396] specifies that the semantics of a relative
   URI reference embedded in a MIME entity is dependent on the base URI.
   The base URI is either (1) the base URI embedded in the MIME entity,
   (2) the base URI of the encapsulating MIME entity, (3) the URI used
   to retrieve the MIME entity, or (4) the application-dependent default
   base URI, where (1) has the highest precedence. [RFC2396] further
   specifies that the mechanism for embedding the base URI is dependent
   on the media type.

   As of today, no established specifications define mechanisms for
   embedding the base URI in XML MIME entities. However, a Proposed
   Recommendation published by W3C, namely "XML Base", attempts to
   define such a mechanism for text/xml, application/xml, text/xml-
   external-parsed-entity, and application/xml-external-parsed-entity.
   The current specification for XML Base is available at
   http://www.w3.org/TR/xmlbase.

7. A Naming Convention for XML-Based Media Types

   This document recommends the use of a naming convention (a suffix of
   '+xml') for identifying XML-based MIME media types, whatever their
   particular content may represent. This allows the use of generic XML
   processors and technologies on a wide variety of different XML
   document types at a minimum cost, using existing frameworks for media
   type registration.

   Although the use of a suffix was not considered as part of the
   original MIME architecture, this choice is considered to provide the
   most functionality with the least potential for interoperability
   problems or lack of future extensibility. The alternatives to the '
   +xml' suffix and the reason for its selection are described in
   Appendix A.

   As XML development continues, new XML document types are appearing
   rapidly. Many of these XML document types would benefit from the
   identification possibilities of a more specific MIME media type than
   text/xml or application/xml can provide, and it is likely that many
   new media types for XML-based document types will be registered in
   the near and ongoing future.

   While the benefits of specific MIME types for particular types of XML
   documents are significant, all XML documents share common structures
   and syntax that make possible common processing.

   Some areas where 'generic' processing is useful include:

   o Browsing - An XML browser can display any XML document with a
      provided [CSS] or [XSLT] style sheet, whatever the vocabulary of
      that document.

   o Editing - Any XML editor can read, modify, and save any XML
      document.

   o Fragment identification - XPointers (work in progress) can work
      with any XML document, whatever vocabulary it uses and whether or
      not it uses XPointer for its own fragment identification.

   o Hypertext linking - XLink (work in progress) hypertext linking is
      designed to connect any XML documents, regardless of vocabulary.

   o Searching - XML-oriented search engines, web crawlers, agents, and
      query tools should be able to read XML documents and extract the
      names and content of elements and attributes even if the tools are
      ignorant of the particular vocabulary used for elements and
      attributes.

   o Storage - XML-oriented storage systems, which keep XML documents
      internally in a parsed form, should similarly be able to process,
      store, and recreate any XML document.

   o Well-formedness and validity checking - An XML processor can
      confirm that any XML document is well-formed and that it is valid
      (i.e., conforms to its declared DTD or Schema).

When a new media type is introduced for an XML-based format, the name
of the media type SHOULD end with '+xml'. This convention will allow
applications that can process XML generically to detect that the MIME
entity is supposed to be an XML document, verify this assumption by
invoking some XML processor, and then process the XML document
accordingly. Applications may match for types that represent XML
MIME entities by comparing the subtype to the pattern '*/*+xml'. (Of
course, 4 of the 5 media types defined in this document -- text/xml,
application/xml, text/xml-external-parsed-entity, and
application/xml-external-parsed-entity -- also represent XML MIME
entities while not conforming to the '*/*+xml' pattern.)

   NOTE: Section 14.1 of HTTP[RFC2616] does not support Accept
   headers of the form "Accept: */*+xml" and so this header MUST NOT
   be used in this way. Instead, content negotiation[RFC2703] could
   potentially be used if an XML-based MIME type were needed.

XML generic processing is not always appropriate for XML-based media
types. For example, authors of some such media types may wish that
the types remain entirely opaque except to applications that are
specifically designed to deal with that media type. By NOT following
the naming convention '+xml', such media types can avoid XML-generic
processing. Since generic processing will be useful in many cases,
however -- including in some situations that are difficult to predict
ahead of time -- those registering media types SHOULD use the '+xml'
convention unless they have a particularly compelling reason not to.

The registration process for these media types is described in
[RFC2048]. The registrar for the IETF tree will encourage new XML-
based media type registrations in the IETF tree to follow this
guideline. Registrars for other trees SHOULD follow this convention

   in order to ensure maximum interoperability of their XML-based
   documents. Similarly, media subtypes that do not represent XML MIME
   entities MUST NOT be allowed to register with a '+xml' suffix.

7.1 Referencing

   Registrations for new XML-based media types under the top-level type
   "text" SHOULD, in specifying the charset parameter and encoding
   considerations, define them as: "Same as [charset parameter /
   encoding considerations] of text/xml as specified in RFC 3023."

   Registrations for new XML-based media types under top-level types
   other than "text" SHOULD, in specifying the charset parameter and
   encoding considerations, define them as: "Same as [charset parameter
   / encoding considerations] of application/xml as specified in RFC
   3023."

   The use of the charset parameter is STRONGLY RECOMMENDED, since this
   information can be used by XML processors to determine
   authoritatively the charset of the XML MIME entity.

   These registrations SHOULD specify that the XML-based media type
   being registered has all of the security considerations described in
   RFC 3023 plus any additional considerations specific to that media
   type.

   These registrations SHOULD also make reference to RFC 3023 in
   specifying magic numbers, fragment identifiers, base URIs, and use of
   the BOM.

   These registrations MAY reference the text/xml registration in RFC
   3023 in specifying interoperability considerations, if these
   considerations are not overridden by issues specific to that media
   type.

8. Examples

   The examples below give the value of the MIME Content-type header and
   the XML declaration (which includes the encoding declaration) inside
   the XML MIME entity. For UTF-16 examples, the Byte Order Mark
   character is denoted as "{BOM}", and the XML declaration is assumed
   to come at the beginning of the XML MIME entity, immediately
   following the BOM. Note that other MIME headers may be present, and
   the XML MIME entity may contain other data in addition to the XML
   declaration; the examples focus on the Content-type header and the
   encoding declaration for clarity.

8.1 Text/xml with UTF-8 Charset

    Content-type: text/xml; charset="utf-8"

    <?xml version="1.0" encoding="utf-8"?>

    This is the recommended charset value for use with text/xml. Since
    the charset parameter is provided, MIME and XML processors MUST treat
    the enclosed entity as UTF-8 encoded.

    If sent using a 7-bit transport (e.g., SMTP[RFC0821]), the XML MIME
    entity MUST use a content-transfer-encoding of either quoted-
    printable or base64. For an 8-bit clean transport (e.g., 8BITMIME
    ESMTP or NNTP), or a binary clean transport (e.g., HTTP), no
    content-transfer-encoding is necessary.

8.2 Text/xml with UTF-16 Charset

    Content-type: text/xml; charset="utf-16"

    {BOM}<?xml version='1.0' encoding='utf-16'?>

    or

    {BOM}<?xml version='1.0'?>

    This is possible only when the XML MIME entity is transmitted via
    HTTP, which uses a MIME-like mechanism and is a binary-clean
    protocol, hence does not perform CR and LF transformations and allows
    NUL octets. As described in [RFC2781], the UTF-16 family MUST NOT be
    used with media types under the top-level type "text" except over
    HTTP (see section 19.4.1 of [RFC2616] for details).

    Since HTTP is binary clean, no content-transfer-encoding is
    necessary.

8.3 Text/xml with UTF-16BE Charset

    Content-type: text/xml; charset="utf-16be"

    <?xml version='1.0' encoding='utf-16be'?>

    Observe that the BOM does not exist. This is again possible only
    when the XML MIME entity is transmitted via HTTP.

8.4 Text/xml with ISO-2022-KR Charset

    Content-type: text/xml; charset="iso-2022-kr"

    <?xml version="1.0" encoding='iso-2022-kr'?>

    This example shows text/xml with a Korean charset (e.g., Hangul)
    encoded following the specification in [RFC1557]. Since the charset
    parameter is provided, MIME and XML processors MUST treat the
    enclosed entity as encoded per RFC 1557.

    Since ISO-2022-KR has been defined to use only 7 bits of data, no
    content-transfer-encoding is necessary with any transport.

8.5 Text/xml with Omitted Charset

    Content-type: text/xml

    {BOM}<?xml version="1.0" encoding="utf-16"?>

    or

    {BOM}<?xml version="1.0"?>

    This example shows text/xml with the charset parameter omitted. In
    this case, MIME and XML processors MUST assume the charset is "us-
    ascii", the default charset value for text media types specified in
    [RFC2046]. The default of "us-ascii" holds even if the text/xml
    entity is transported using HTTP.

    Omitting the charset parameter is NOT RECOMMENDED for text/xml. For
    example, even if the contents of the XML MIME entity are UTF-16 or
    UTF-8, or the XML MIME entity has an explicit encoding declaration,
    XML and MIME processors MUST assume the charset is "us-ascii".

8.6 Application/xml with UTF-16 Charset

    Content-type: application/xml; charset="utf-16"

    {BOM}<?xml version="1.0" encoding="utf-16"?>

    or

    {BOM}<?xml version="1.0"?>

    This is a recommended charset value for use with application/xml.
    Since the charset parameter is provided, MIME and XML processors MUST
    treat the enclosed entity as UTF-16 encoded.

If sent using a 7-bit transport (e.g., SMTP) or an 8-bit clean
transport (e.g., 8BITMIME ESMTP or NNTP), the XML MIME entity MUST be
encoded in quoted-printable or base64. For a binary clean transport
(e.g., HTTP), no content-transfer-encoding is necessary.

8.7 Application/xml with UTF-16BE Charset

    Content-type: application/xml; charset="utf-16be"

    <?xml version='1.0' encoding='utf-16be'?>

    Observe that the BOM does not exist. Since the charset parameter is
    provided, MIME and XML processors MUST treat the enclosed entity as
    UTF-16BE encoded.

8.8 Application/xml with ISO-2022-KR Charset

    Content-type: application/xml; charset="iso-2022-kr"

    <?xml version="1.0" encoding="iso-2022-kr"?>

    This example shows application/xml with a Korean charset (e.g.,
    Hangul) encoded following the specification in [RFC1557]. Since the
    charset parameter is provided, MIME and XML processors MUST treat the
    enclosed entity as encoded per RFC 1557, independent of whether the
    XML MIME entity has an internal encoding declaration (this example
    does show such a declaration, which agrees with the charset
    parameter).

    Since ISO-2022-KR has been defined to use only 7 bits of data, no
    content-transfer-encoding is necessary with any transport.

8.9 Application/xml with Omitted Charset and UTF-16 XML MIME Entity

    Content-type: application/xml

    {BOM}<?xml version='1.0' encoding="utf-16"?>

    or

    {BOM}<?xml version='1.0'?>

    For this example, the XML MIME entity begins with a BOM. Since the
    charset has been omitted, a conforming XML processor follows the
    requirements of [XML], section 4.3.3. Specifically, the XML
    processor reads the BOM, and thus knows deterministically that the
    charset is UTF-16.

An XML-unaware MIME processor SHOULD make no assumptions about the
charset of the XML MIME entity.

8.10 Application/xml with Omitted Charset and UTF-8 Entity

   Content-type: application/xml

   <?xml version='1.0'?>

   In this example, the charset parameter has been omitted, and there is
   no BOM. Since there is no BOM, the XML processor follows the
   requirements in section 4.3.3 of [XML], and optionally applies the
   mechanism described in Appendix F (which is non-normative) of [XML]
   to determine the charset encoding of UTF-8. The XML MIME entity does
   not contain an encoding declaration, but since the encoding is UTF-8,
   this is still a conforming XML MIME entity.

   An XML-unaware MIME processor SHOULD make no assumptions about the
   charset of the XML MIME entity.

8.11 Application/xml with Omitted Charset and Internal Encoding
     Declaration

   Content-type: application/xml

   <?xml version='1.0' encoding="iso-10646-ucs-4"?>

   In this example, the charset parameter has been omitted, and there is
   no BOM. However, the XML MIME entity does have an encoding
   declaration inside the XML MIME entity that specifies the entity's
   charset. Following the requirements in section 4.3.3 of [XML], and
   optionally applying the mechanism described in Appendix F (non-
   normative) of [XML], the XML processor determines the charset of the
   XML MIME entity (in this example, UCS-4).

   An XML-unaware MIME processor SHOULD make no assumptions about the
   charset of the XML MIME entity.

8.12 Text/xml-external-parsed-entity with UTF-8 Charset

   Content-type: text/xml-external-parsed-entity; charset="utf-8"

   <?xml encoding="utf-8"?>

   This is the recommended charset value for use with text/xml-
   external-parsed-entity. Since the charset parameter is provided,
   MIME and XML processors MUST treat the enclosed entity as UTF-8
   encoded.

If sent using a 7-bit transport (e.g., SMTP), the XML MIME entity
MUST use a content-transfer-encoding of either quoted-printable or
base64. For an 8-bit clean transport (e.g., 8BITMIME ESMTP or NNTP),
or a binary clean transport (e.g., HTTP) no content-transfer-encoding
is necessary.

8.13 Application/xml-external-parsed-entity with UTF-16 Charset

    Content-type: application/xml-external-parsed-entity;
     charset="utf-16"

    {BOM}<?xml encoding="utf-16"?>

    or

    {BOM}<?xml?>

    This is a recommended charset value for use with application/xml-
    external-parsed-entity. Since the charset parameter is provided,
    MIME and XML processors MUST treat the enclosed entity as UTF-16
    encoded.

    If sent using a 7-bit transport (e.g., SMTP) or an 8-bit clean
    transport (e.g., 8BITMIME ESMTP or NNTP), the XML MIME entity MUST be
    encoded in quoted-printable or base64. For a binary clean transport
    (e.g., HTTP), no content-transfer-encoding is necessary.

8.14 Application/xml-external-parsed-entity with UTF-16BE Charset

    Content-type: application/xml-external-parsed-entity;
     charset="utf-16be"

    <?xml encoding="utf-16be"?>

    Since the charset parameter is provided, MIME and XML processors MUST
    treat the enclosed entity as UTF-16BE encoded.

8.15 Application/xml-dtd

    Content-type: application/xml-dtd; charset="utf-8"

    <?xml encoding="utf-8"?>

    Charset "utf-8" is a recommended charset value for use with
    application/xml-dtd. Since the charset parameter is provided, MIME
    and XML processors MUST treat the enclosed entity as UTF-8 encoded.

8.16 Application/mathml+xml

    Content-type: application/mathml+xml

    <?xml version="1.0" ?>

    MathML documents are XML documents whose content describes
    mathematical information, as defined by [MathML]. As a format based
    on XML, MathML documents SHOULD use the '+xml' suffix convention in
    their MIME content-type identifier. However, no content type has yet
    been registered for MathML and so this media type should not be used
    until such registration has been completed.

8.17 Application/xslt+xml

    Content-type: application/xslt+xml

    <?xml version="1.0" ?>

    Extensible Stylesheet Language (XSLT) documents are XML documents
    whose content describes stylesheets for other XML documents, as
    defined by [XSLT]. As a format based on XML, XSLT documents SHOULD
    use the '+xml' suffix convention in their MIME content-type
    identifier. However, no content type has yet been registered for
    XSLT and so this media type should not be used until such
    registration has been completed.

8.18 Application/rdf+xml

    Content-type: application/rdf+xml

    <?xml version="1.0" ?>

    RDF documents identified using this MIME type are XML documents whose
    content describes metadata, as defined by [RDF]. As a format based
    on XML, RDF documents SHOULD use the '+xml' suffix convention in
    their MIME content-type identifier. However, no content type has yet
    been registered for RDF and so this media type should not be used
    until such registration has been completed.

8.19 Image/svg+xml

    Content-type: image/svg+xml

    <?xml version="1.0" ?>

Scalable Vector Graphics (SVG) documents are XML documents whose
content describes graphical information, as defined by [SVG]. As a
format based on XML, SVG documents SHOULD use the '+xml' suffix
convention in their MIME content-type identifier. However, no
content type has yet been registered for SVG and so this media type
should not be used until such registration has been completed.

8.20 INCONSISTENT EXAMPLE: Text/xml with UTF-8 Charset

    Content-type: text/xml; charset="utf-8"

    <?xml version="1.0" encoding="iso-8859-1"?>

    Since the charset parameter is provided in the Content-Type header,
    MIME and XML processors MUST treat the enclosed entity as UTF-8
    encoded. That is, the "iso-8859-1" encoding MUST be ignored.

    Processors generating XML MIME entities MUST NOT label conflicting
    charset information between the MIME Content-Type and the XML
    declaration.

9. IANA Considerations

    As described in Section 7, this document updates the [RFC2048]
    registration process for XML-based MIME types.

10. Security Considerations

    XML, as a subset of SGML, has all of the same security considerations
    as specified in [RFC1874], and likely more, due to its expected
    ubiquitous deployment.

    To paraphrase section 3 of RFC 1874, XML MIME entities contain
    information to be parsed and processed by the recipient's XML system.
    These entities may contain and such systems may permit explicit
    system level commands to be executed while processing the data. To
    the extent that an XML system will execute arbitrary command strings,
    recipients of XML MIME entities may be a risk. In general, it may be
    possible to specify commands that perform unauthorized file
    operations or make changes to the display processor's environment
    that affect subsequent operations.

    In general, any information stored outside of the direct control of
    the user -- including CSS style sheets, XSL transformations, entity
    declarations, and DTDs -- can be a source of insecurity, by either
    obvious or subtle means. For example, a tiny "whiteout attack"
    modification made to a "master" style sheet could make words in
    critical locations disappear in user documents, without directly

   modifying the user document or the stylesheet it references. Thus,
   the security of any XML document is vitally dependent on all of the
   documents recursively referenced by that document.

   The entity lists and DTDs for XHTML 1.0[XHTML], for instance, are
   likely to be a commonly used set of information. Many developers
   will use and trust them, few of whom will know much about the level
   of security on the W3C's servers, or on any similarly trusted
   repository.

   The simplest attack involves adding declarations that break
   validation. Adding extraneous declarations to a list of character
   entities can effectively "break the contract" used by documents. A
   tiny change that produces a fatal error in a DTD could halt XML
   processing on a large scale. Extraneous declarations are fairly
   obvious, but more sophisticated tricks, like changing attributes from
   being optional to required, can be difficult to track down. Perhaps
   the most dangerous option available to crackers is redefining default
   values for attributes: e.g., if developers have relied on defaulted
   attributes for security, a relatively small change might expose
   enormous quantities of information.

   Apart from the structural possibilities, another option, "entity
   spoofing," can be used to insert text into documents, vandalizing and
   perhaps conveying an unintended message. Because XML 1.0 permits
   multiple entity declarations, and the first declaration takes
   precedence, it's possible to insert malicious content where an entity
   is used, such as by inserting the full text of Winnie the Pooh in
   every occurrence of —.

   Use of the digital signatures work currently underway by the xmldsig
   working group may eventually ameliorate the dangers of referencing
   external documents not under one's own control.

   Use of XML is expected to be varied, and widespread. XML is under
   scrutiny by a wide range of communities for use as a common syntax
   for community-specific metadata. For example, the Dublin
   Core[RFC2413] group is using XML for document metadata, and a new
   effort has begun that is considering use of XML for medical
   information. Other groups view XML as a mechanism for marshalling
   parameters for remote procedure calls. More uses of XML will
   undoubtedly arise.

   Security considerations will vary by domain of use. For example, XML
   medical records will have much more stringent privacy and security
   considerations than XML library metadata. Similarly, use of XML as a
   parameter marshalling syntax necessitates a case by case security
   review.

XML may also have some of the same security concerns as plain text.
Like plain text, XML can contain escape sequences that, when
displayed, have the potential to change the display processor
environment in ways that adversely affect subsequent operations.
Possible effects include, but are not limited to, locking the
keyboard, changing display parameters so subsequent displayed text is
unreadable, or even changing display parameters to deliberately
obscure or distort subsequent displayed material so that its meaning
is lost or altered. Display processors SHOULD either filter such
material from displayed text or else make sure to reset all important
settings after a given display operation is complete.

Some terminal devices have keys whose output, when pressed, can be
changed by sending the display processor a character sequence. If
this is possible the display of a text object containing such
character sequences could reprogram keys to perform some illicit or
dangerous action when the key is subsequently pressed by the user.
In some cases not only can keys be programmed, they can be triggered
remotely, making it possible for a text display operation to directly
perform some unwanted action. As such, the ability to program keys
SHOULD be blocked either by filtering or by disabling the ability to
program keys entirely.

Note that it is also possible to construct XML documents that make
use of what XML terms "entity references" (using the XML meaning of
the term "entity" as described in Section 2), to construct repeated
expansions of text. Recursive expansions are prohibited by [XML] and
XML processors are required to detect them. However, even non-
recursive expansions may cause problems with the finite computing
resources of computers, if they are performed many times.

References

   [ASCII]   "US-ASCII. Coded Character Set -- 7-Bit American Standard
              Code for Information Interchange", ANSI X3.4-1986, 1986.

   [CSS]   Bos, B., Lie, H.W., Lilley, C. and I. Jacobs, "Cascading
              Style Sheets, level 2 (CSS2) Specification", World Wide
              Web Consortium Recommendation REC-CSS2, May 1998,
              <http://www.w3.org/TR/REC-CSS2/>.

   [ISO8859] "ISO-8859. International Standard -- Information
              Processing -- 8-bit Single-Byte Coded Graphic Character
              Sets -- Part 1: Latin alphabet No. 1, ISO-8859-1:1987",
              1987.

   [MathML]   Ion, P. and R. Miner, "Mathematical Markup Language
               (MathML) 1.01", World Wide Web Consortium Recommendation
               REC-MathML, July 1999, <http://www.w3.org/TR/REC-MathML/>.

   [PNG]   Boutell, T., "PNG (Portable Network Graphics)
               Specification", World Wide Web Consortium Recommendation
               REC-png, October 1996, <http://www.w3.org/TR/REC-png>.

   [RDF]   Lassila, O. and R.R. Swick, "Resource Description
               Framework (RDF) Model and Syntax Specification", World
               Wide Web Consortium Recommendation REC-rdf-syntax,
               February 1999, <http://www.w3.org/TR/REC-rdf-syntax/>.

   [RFC0821]   Postel, J., "Simple Mail Transfer Protocol", STD 10, RFC
               821, August 1982.

   [RFC0977]   Kantor, B. and P. Lapsley, "Network News Transfer
               Protocol", RFC 977, February 1986.

   [RFC1557]   Choi, U., Chon, K. and H. Park, "Korean Character Encoding
               for Internet Messages", RFC 1557, December 1993.

   [RFC1652]   Klensin, J., Freed, N., Rose, M., Stefferud, E. and D.
               Crocker, "SMTP Service Extension for 8bit-MIMEtransport",
               RFC 1652, July 1994.

   [RFC1874]   Levinson, E., "SGML Media Types", RFC 1874, December 1995.

   [RFC2045]   Freed, N. and N. Borenstein, "Multipurpose Internet Mail
               Extensions (MIME) Part One: Format of Internet Message
               Bodies", RFC 2045, November 1996.

   [RFC2046]   Freed, N. and N. Borenstein, "Multipurpose Internet Mail
               Extensions (MIME) Part Two: Media Types", RFC 2046,
               November 1996.

   [RFC2048]   Freed, N., Klensin, J. and J. Postel, "Multipurpose
               Internet Mail Extensions (MIME) Part Four: Registration
               Procedures", RFC 2048, November 1996.

   [RFC2060]   Crispin, M., "Internet Message Access Protocol - Version
               4rev1", RFC 2060, December 1996.

   [RFC2077]   Nelson, S., Parks, C. and Mitra, "The Model Primary
               Content Type for Multipurpose Internet Mail Extensions",
               RFC 2077, January 1997.

   [RFC2119]   Bradner, S., "Key words for use in RFCs to Indicate
               Requirement Levels", BCP 14, RFC 2119, March 1997.

   [RFC2130]   Weider, C., Preston, C., Simonsen, K., Alvestrand, H.,
               Atkinson, R., Crispin, M. and P. Svanberg, "The Report of
               the IAB Character Set Workshop held 29 February - 1 March,
               1996", RFC 2130, April 1997.

   [RFC2279]   Yergeau, F., "UTF-8, a transformation format of ISO
               10646", RFC 2279, January 1998.

   [RFC2376]   Whitehead, E. and M. Murata, "XML Media Types", RFC 2376,
               July 1998.

   [RFC2396]   Berners-Lee, T., Fielding, R. and L. Masinter, "Uniform
               Resource Identifiers (URI): Generic Syntax.", RFC 2396,
               August 1998.

   [RFC2413]   Weibel, S., Kunze, J., Lagoze, C. and M. Wolf, "Dublin
               Core Metadata for Resource Discovery", RFC 2413, September
               1998.

   [RFC2445]   Dawson, F. and D. Stenerson, "Internet Calendaring and
               Scheduling Core Object Specification (iCalendar)", RFC
               2445, November 1998.

   [RFC2518]   Goland, Y., Whitehead, E., Faizi, A., Carter, S. and D.
               Jensen, "HTTP Extensions for Distributed Authoring --
               WEBDAV", RFC 2518, February 1999.

   [RFC2616]   Fielding, R., Gettys, J., Mogul, J., Nielsen, H.,
               Masinter, L., Leach, P. and T. Berners-Lee, "Hypertext
               Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

   [RFC2629]   Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629,
               June 1999.

   [RFC2703]   Klyne, G., "Protocol-independent Content Negotiation
               Framework", RFC 2703, September 1999.

   [RFC2781]   Hoffman, P. and F. Yergeau, "UTF-16, an encoding of ISO
               10646", RFC 2781, Februrary 2000.

   [RFC2801]   Burdett, D., "Internet Open Trading Protocol - IOTP
               Version 1.0", RFC 2801, April 2000.

   [SGML]   International Standard Organization, "Information
                 Processing -- Text and Office Systems -- Standard
                 Generalized Markup Language (SGML)", ISO 8879, October
                 1986.

   [SVG]   Ferraiolo, J., "Scalable Vector Graphics (SVG)", World
                 Wide Web Consortium Candidate Recommendation SVG, November
                 2000, <http://www.w3.org/TR/SVG>.

   [XHTML]   Pemberton, S. and et al, "XHTML 1.0: The Extensible
                 HyperText Markup Language", World Wide Web Consortium
                 Recommendation xhtml1, January 2000,
                 <http://www.w3.org/TR/xhtml1>.

   [XML]   Bray, T., Paoli, J., Sperberg-McQueen, C.M. and E. Maler,
                 "Extensible Markup Language (XML) 1.0 (Second Edition)",
                 World Wide Web Consortium Recommendation REC-xml, October
                 2000, <http://www.w3.org/TR/REC-xml>.

   [XSLT]   Clark, J., "XSL Transformations (XSLT) Version 1.0", World
                 Wide Web Consortium Recommendation xslt, November 1999,
                 <http://www.w3.org/TR/xslt>.

Authors' Addresses

    MURATA Makoto (FAMILY Given)
    IBM Tokyo Research Laboratory
    1623-14, Shimotsuruma
    Yamato-shi, Kanagawa-ken 242-8502
    Japan

    Phone: +81-46-215-4678
    EMail: mmurata@trl.ibm.co.jp

    Simon St.Laurent
    simonstl.com
    1259 Dryden Road
    Ithaca, New York 14850
    USA

    EMail: simonstl@simonstl.com
    URI: http://www.simonstl.com/

    Dan Kohn
    Skymoon Ventures
    3045 Park Boulevard
    Palo Alto, California 94306
    USA

    Phone: +1-650-327-2600
    EMail: dan@dankohn.com
    URI: http://www.dankohn.com/

Appendix A. Why Use the '+xml' Suffix for XML-Based MIME Types?

   Although the use of a suffix was not considered as part of the
   original MIME architecture, this choice is considered to provide the
   most functionality with the least potential for interoperability
   problems or lack of future extensibility. The alternatives to the
   '+xml' suffix and the reason for its selection are described below.

A.1 Why not just use text/xml or application/xml and let the XML
     processor dispatch to the correct application based on the
     referenced DTD?

   text/xml and application/xml remain useful in many situations,
   especially for document-oriented applications that involve combining
   XML with a stylesheet in order to present the data. However, XML is
   also used to define entirely new data types, and an XML-based format
   such as image/svg+xml fits the definition of a MIME media type
   exactly as well as image/png[PNG] does. (Note that image/svg+xml is
   not yet registered.) Although extra functionality is available for
   MIME processors that are also XML processors, XML-based media types
   -- even when treated as opaque, non-XML media types -- are just as
   useful as any other media type and should be treated as such.

   Since MIME dispatchers work off of the MIME type, use of text/xml or
   application/xml to label discrete media types will hinder correct
   dispatching and general interoperability. Finally, many XML
   documents use neither DTDs nor namespaces, yet are perfectly legal
   XML.

A.2 Why not create a new subtree (e.g., image/xml.svg) to represent XML
     MIME types?

   The subtree under which a media type is registered -- IETF, vendor
   (*/vnd.*), or personal (*/prs.*); see [RFC2048] for details -- is
   completely orthogonal from whether the media type uses XML syntax or
   not. The suffix approach allows XML document types to be identified
   within any subtree. The vendor subtree, for example, is likely to
   include a large number of XML-based document types. By using a
   suffix, rather than setting up a separate subtree, those types may
   remain in the same location in the tree of MIME types that they would
   have occupied had they not been based on XML.

A.3 Why not create a new top-level MIME type for XML-based media types?

   The top-level MIME type (e.g., model/*[RFC2077]) determines what kind
   of content the type is, not what syntax it uses. For example, agents
   using image/* to signal acceptance of any image format should
   certainly be given access to media type image/svg+xml, which is in

all respects a standard image subtype. It just happens to use XML to
describe its syntax. The two aspects of the media type are
completely orthogonal.

XML-based data types will most likely be registered in ALL top-level
categories. Potential, though currently unregistered, examples could
include application/mathml+xml[MathML] and image/svg+xml[SVG].

A.4 Why not just have the MIME processor 'sniff' the content to
    determine whether it is XML?

Rather than explicitly labeling XML-based media types, the processor
could look inside each type and see whether or not it is XML. The
processor could also cache a list of XML-based media types.

Although this method might work acceptably for some mail
applications, it would fail completely in many other uses of MIME.
For instance, an XML-based web crawler would have no way of
determining whether a file is XML except to fetch it and check. The
same issue applies in some IMAP4[RFC2060] mail applications, where
the client first fetches the MIME type as part of the message
structure and then decides whether to fetch the MIME entity.
Requiring these fetches just to determine whether the MIME type is
XML could have significant bandwidth and latency disadvantages in
many situations.

Sniffing XML also isn't as simple as it might seem. DOCTYPE
declarations aren't required, and they can appear fairly deep into a
document under certain unpreventable circumstances. (E.g., the XML
declaration, comments, and processing instructions can occupy space
before the DOCTYPE declaration.) Even sniffing the DOCTYPE isn't
completely reliable, thanks to a variety of issues involving default
values for namespaces within external DTDs and overrides inside the
internal DTD. Finally, the variety in potential character encodings
(something XML provides tools to deal with), also makes reliable
sniffing less likely.

A.5 Why not use a MIME parameter to specify that a media type uses XML
    syntax?

For example, one could use "Content-Type: application/iotp;
alternate-type=text/xml" or "Content-Type: application/iotp;
syntax=xml".

Section 5 of [RFC2045] says that "Parameters are modifiers of the
media subtype, and as such do not fundamentally affect the nature of
the content". However, all XML-based media types are by their nature

always XML. Parameters, as they have been defined in the MIME
architecture, are never invariant across all instantiations of a
media type.

More practically, very few if any MIME dispatchers and other MIME
agents support dispatching off of a parameter. While MIME agents on
the receiving side will need to be updated in either case to support
(or fall back to) generic XML processing, it has been suggested that
it is easier to implement this functionality when acting off of the
media type rather than a parameter. More important, sending agents
require no update to properly tag an image as "image/svg+xml", but
few if any sending agents currently support always tagging certain
content types with a parameter.

A.6 How about labeling with parameters in the other direction (e.g.,
    application/xml; Content-Feature=iotp)?

This proposal fails under the simplest case, of a user with neither
knowledge of XML nor an XML-capable MIME dispatcher. In that case,
the user's MIME dispatcher is likely to dispatch the content to an
XML processing application when the correct default behavior should
be to dispatch the content to the application responsible for the
content type (e.g., an ecommerce engine for
application/iotp+xml[RFC2801], once this media type is registered).

Note that even if the user had already installed the appropriate
application (e.g., the ecommerce engine), and that installation had
updated the MIME registry, many operating system level MIME
registries such as .mailcap in Unix and HKEY_CLASSES_ROOT in Windows
do not currently support dispatching off a parameter, and cannot
easily be upgraded to do so. And, even if the operating system were
upgraded to support this, each MIME dispatcher would also separately
need to be upgraded.

A.7 How about a new superclass MIME parameter that is defined to apply
    to all MIME types (e.g., Content-Type: application/iotp;
    $superclass=xml)?

This combines the problems of Appendix A.5 and Appendix A.6.

If the sender attaches an image/svg+xml file to a message and
includes the instructions "Please copy the French text on the road
sign", someone with an XML-aware MIME client and an XML browser but
no support for SVG can still probably open the file and copy the
text. By contrast, with superclasses, the sender must add superclass
support to her existing mailer AND the receiver must add superclass
support to his before this transaction can work correctly.

If the receiver comes to rely on the superclass tag being present and
applications are deployed relying on that tag (as always seems to
happen), then only upgraded senders will be able to interoperate with
those receiving applications.

A.8 What about adding a new parameter to the Content-Disposition header
     or creating a new Content-Structure header to indicate XML syntax?

This has nearly identical problems to Appendix A.7, in that it
requires both senders and receivers to be upgraded, and few if any
operating systems and MIME dispatchers support working off of
anything other than the MIME type.

A.9 How about a new Alternative-Content-Type header?

This is better than Appendix A.8, in that no extra functionality
needs to be added to a MIME registry to support dispatching of
information other than standard content types. However, it still
requires both sender and receiver to be upgraded, and it will also
fail in many cases (e.g., web hosting to an outsourced server), where
the user can set MIME types (often through implicit mapping to file
extensions), but has no way of adding arbitrary HTTP headers.

A.10 How about using a conneg tag instead (e.g., accept-features:
     (syntax=xml))?

When the conneg protocol is fully defined, this may potentially be a
reasonable thing to do. But given the limited current state of
conneg[RFC2703] development, it is not a credible replacement for a
MIME-based solution.

A.11 How about a third-level content-type, such as text/xml/rdf?

MIME explicitly defines two levels of content type, the top-level for
the kind of content and the second-level for the specific media type.
[RFC2048] extends this in an interoperable way by using prefixes to
specify separate trees for IETF, vendor, and personal registrations.
This specification also extends the two-level type by using the '
+xml' suffix. In both cases, processors that are unaware of these
later specifications treat them as opaque and continue to
interoperate. By contrast, adding a third-level type would break the
current MIME architecture and cause numerous interoperability
failures.

A.12 Why use the plus ('+') character for the suffix '+xml'?

   As specified in Section 5.1 of [RFC2045], a tspecial can't be used:

      tspecials :=
      "(" / ")" / "<" / ">" / "@" /
      "," / ";" / ":" / "\" / <">
      "/" / "[" / "]" / "?" / "="

   It was thought that "." would not be a good choice since it is
   already used as an additional hierarchy delimiter. Also, "*" has a
   common wildcard meaning, and "-" and "_" are common word separators
   and easily confused. The characters %'`#& are frequently used for
   quoting or comments and so are not ideal.

   That leaves: ~!$^+{}|

   Note that "-" is used heavily in the current registry. "$" and "_"
   are used once each. The others are currently unused.

   It was thought that '+' expressed the semantics that a MIME type can
   be treated (for example) as both scalable vector graphics AND ALSO as
   XML; it is both simultaneously.

A.13 What is the semantic difference between application/foo and
      application/foo+xml?

   MIME processors that are unaware of XML will treat the '+xml' suffix
   as completely opaque, so it is essential that no extra semantics be
   assigned to its presence. Therefore, application/foo and
   application/foo+xml SHOULD be treated as completely independent media
   types. Although, for example, text/calendar+xml could be an XML
   version of text/calendar[RFC2445], it is possible that this
   (hypothetical) new media type would include new semantics as well as
   new syntax, and in any case, there would be many applications that
   support text/calendar but had not yet been upgraded to support
   text/calendar+xml.

A.14 What happens when an even better markup language (e.g., EBML) is
      defined, or a new category of data?

   In the ten years that MIME has existed, XML is the first generic data
   format that has seemed to justify special treatment, so it is hoped
   that no further suffixes will be necessary. However, if some are
   later defined, and these documents were also XML, they would need to
   specify that the '+xml' suffix is always the outermost suffix (e.g.,

application/foo+ebml+xml not application/foo+xml+ebml). If they were
not XML, then they would use a regular suffix (e.g.,
application/foo+ebml).

A.15 Why must I use the '+xml' suffix for my new XML-based media type?

   You don't have to, but unless you have a good reason to explicitly
   disallow generic XML processing, you should use the suffix so as not
   to curtail the options of future users and developers.

   Whether the inventors of a media type, today, design it for dispatch
   to generic XML processing machinery (and most won't) is not the
   critical issue. The core notion is that the knowledge that some
   media type happens to use XML syntax opens the door to unanticipated
   kinds of processing beyond those envisioned by its inventors, and on
   this basis identifying such encoding is a good and useful thing.

   Developers of new media types are often tightly focused on a
   particular type of processing that meets current needs. But there is
   no need to rule out generic processing as well, which could make your
   media type more valuable over time. It is believed that registering
   with the '+xml' suffix will cause no interoperability problems
   whatsoever, while it may enable significant new functionality and
   interoperability now and in the future. So, the conservative
   approach is to include the '+xml' suffix.

Appendix B. Changes from RFC 2376

   There are numerous and significant differences between this
   specification and [RFC2376], which it obsoletes. This appendix
   summarizes the major differences only.

   First, text/xml-external-parsed-entity and application/xml-external-
   parsed-entity are added as media types for external parsed entities,
   and text/xml and application/xml are now prohibited.

   Second, application/xml-dtd is added as a media type for external DTD
   subsets and external parameter entities, and text/xml and
   application/xml are now prohibited.

   Third, "utf-16le" and "utf-16be" are added. RFC 2781 has introduced
   these BOM-less variations of the UTF-16 family.

   Fourth, a naming convention ('+xml') for XML-based media types has
   been added, which also updates [RFC2048] as described in Section 7.
   By following this convention, an XML-based media type can be easily
   recognized as such.

Appendix C. Acknowledgements

    This document reflects the input of numerous participants to the
    ietf-xml-mime@imc.org mailing list, though any errors are the
    responsibility of the authors. Special thanks to:

    Mark Baker, James Clark, Dan Connolly, Martin Duerst, Ned Freed,
    Yaron Goland, Rick Jelliffe, Larry Masinter, David Megginson, Keith
    Moore, Chris Newman, Gavin Nicol, Marshall Rose, Jim Whitehead and
    participants of the XML activity at the W3C.

Full Copyright Statement

Acknowledgement